Remarks

In the present response, ten claims (1-3, 6-9, 12, 14, and 17). Claims 1-17 are presented for examination. Applicants believe that no new matter is entered.

I. Claim Rejections: 35 USC § 102

Claims 1, 3-7, 10-13, and 15-17 are rejected under 35 USC § 102 as being anticipated by Sterling et al. (USPN 5,822,588, hereinafter Sterling). This rejection is traversed.

A proper rejection of a claim under 35 U.S.C. §102 requires that a single prior art reference disclose each element of the claim. See MPEP § 2131, also, *W.L. Gore & Assoc., Inc. v. Garlock, Inc.*, 721 F.2d 1540, 220 U.S.P.Q. 303, 313 (Fed. Cir. 1983). Since Sterling neither teaches nor suggests each element in claims 1-17, these claims are allowable over Sterling.

Claim 1

Independent claim 1 recites numerous limitations that are not taught or suggested in Sterling. For example, claim 1 recites "spawning, by the computer program, a plurality of threads that are capable of being executed concurrently." By contrast, Sterling does not teach or suggest a method for spawning a plurality of threads as claimed.

As another example, claim 1 recites "verifying the validity of the thread-local annotation if the element is annotated as thread-local **by determining whether the element annotated as thread-local can be accessed through more than one thread**." This limitation is not taught or suggested in Sterling. By contrast, Sterling teaches how the analyze operation works:

> In the preferred embodiment, the "analyze" routine then simulates the execution of the target program code. This is done by using the flow information generated in the previous step, and following the movement of the target program through the basic blocks. For each basic block, the activity for that block is simulated. For each part of that activity, the following actions are taken;

> If a lock's state is changed (e.g. the lock is acquired or
>
> released), check to see whether the change is legal given the current state of the lock. For example, releasing a lock which is not held is an error. When a lock is acquired, if other locks are already held, record the orderings in the adjacency matrix and check to make sure the order does not violate those specified in annotations. Check to make sure that if a covering lock has been defined for the lock, the covering lock is always held whenever the covered lock is held.
>
> check any "assertions" (for example, if it was asserted that "mutex x is held" at this point, check the state of lock X to see if it is "held");
>
> if a variable is accessed, and an annotation specified which lock protects that variable, check to see if that lock is "held". If the variable was designated read-only, check to see that it is not being written. In any case, update the list of locks consistently held when that variable is accessed;
>
> for a function call, recursively simulate the called function, noting the state of any locks involved as indicated above. (Col. 19, line 48 – Col. 20, line 9).

Dependent claims 4 and 10-11 depend from claim 1 and thus inherit all the limitations of base claim 1. As such, claims 4 and 10-11 are also allowable over Sterling. Further, these dependent claims contain numerous limitations not taught or suggested in Sterling.

Claim 3

Independent claim 3 recites numerous limitations that are not taught or suggested in Sterling. For example, claim 3 recites "spawning a plurality of threads that concurrently run." By contrast, Sterling does not teach or suggest a method for spawning a plurality of threads as claimed.

As another example, claim 3 recites "verifying the validity of the element annotated as thread-local or thread-shared **by determining if an element annotated as**

thread-shared includes a portion annotated as thread-local and/or a link to another element that is annotated as thread-local." This limitation is not taught or suggested in Sterling. By contrast, Sterling teaches how the analyze operation works (see quotation above in connection with Claim 1), but the claimed limitation is not discussed or suggested.

Dependent claim 5 depends from claim 3 and thus inherits all the limitations of base claim 3. As such, claim 5 is also allowable over Sterling. Further, this dependent claim contains numerous limitations not taught or suggested in Sterling.

<u>**Claim 6**</u>

Independent claim 6 recites numerous limitations that are not taught or suggested in Sterling. For example, claim 6 recites "spawning a plurality of threads that concurrently run." By contrast, Sterling does not teach or suggest a method for spawning a plurality of threads as claimed.

As another example, claim 6 recites "verifying the validity of the element annotated as thread-local or thread-shared **by determining if an element annotated as thread-shared includes a pointer or a reference to a different element.**" This limitation is not taught or suggested in Sterling. By contrast, Sterling teaches how the analyze operation works (see quotation above in connection with Claim 1), but the claimed limitation is not discussed or suggested.

Dependent claim 7 depends from claim 6 and thus inherits all the limitations of base claim 6. As such, claim 7 is also allowable over Sterling. Further, this dependent claim contains numerous limitations not taught or suggested in Sterling.

<u>**Claim 12**</u>

Independent claim 12 recites numerous limitations that are not taught or suggested in Sterling. For example, claim 12 recites "means for verifying the validity of the thread-local annotation if the element is annotated as thread-local **by determining whether the element annotated as thread-local can be accessed from a thread-**

shared element rather than by one and only one thread." This limitation is not taught or suggested in Sterling. By contrast, Sterling teaches how the analyze operation works (see quotation above in connection with Claim 1), but the claimed limitation is not discussed or suggested.

As yet another example, claim 12 recites "means for providing a warning **if the element annotated as thread-local can be accessed from a thread-shared element.**" This limitation is not taught or suggested in Sterling. By contrast, Sterling teaches:

> In one of those paths the program unlocks the lock without having locked it--clearly an error--and in the other the program leaves the lock locked, causing the function to appear to have inconsistent side-effects on the locks. When warlock encounters inconsistent side effects like this, it warns the user and eliminates from its analysis all control paths through the function which lead to incorrect side effects. (Col. 22, lines 3-1).

Thus, Sterling teaches issuing a warning when the program unlocks the lock without having locked it. Claim 12, however, recites providing a warning if the element annotated as thread-local can be accessed from a thread-shared element.

Dependent claims 13 and 15-16 depend from claim 12 and thus inherit all the limitations of base claim 12. As such, claims 13 and 15-16 are also allowable over Sterling. Further, these dependent claims contain numerous limitations not taught or suggested in Sterling.

Claim 17

Independent claim 17 recites numerous limitations that are not taught or suggested in Sterling. For example, claim 17 recites "verify the validity of the element annotated as thread-local or thread-shared **by determining when a second thread is spawned from a first thread that any elements passed from the first thread during creation of the second thread are annotated as being thread-shared.**" This limitation is not taught or suggested in Sterling. By contrast, Sterling teaches how the analyze operation works (see quotation above in connection with Claim 1), but the claimed limitation is not discussed or suggested.

II. Allowable Claims

Applicants sincerely thank the Examiner for allowing claims 2, 8, 9, and 14. These claims are placed in independent form including all the limitations of the base claim and any intervening claims.

CONCLUSION

In view of the above, Applicant believes claims 1-17 are in condition for allowance. Allowance of these claims is respectfully requested.

Any inquiry regarding this Amendment and Response should be directed to Philip S. Lyren at Telephone No. (281) 514-8236, Facsimile No. (281) 514-8332. In addition, all correspondence should continue to be directed to the following address:

Hewlett-Packard Company
Intellectual Property Administration
P.O. Box 272400
Fort Collins, Colorado 80527-2400

Respectfully submitted,

[illegible]

Philip S. Lyren
Reg. No. 40,709
Ph: 281-514-8236

CERTIFICATE UNDER 37 C.F.R. 1.8: The undersigned hereby certifies that this paper or papers, as described herein, are being deposited in the United States Postal Service, as first class mail, in an envelope address to: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450 on this 14th day of June, 2004.

By Be Henry
Name: Be Henry